Vol 2
AΦ40828

12

RADC-TR-77-162, Volume I (of two)
Final Technical Report
May 1977

IMPACT OF STRUCTURED PROGRAMMING STANDARDS ON SMALL GOVERNMENT CONTRACTORS

W. W. Gaertner Research Inc

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

APPROVED: [illegible]

MICHAEL LANDES
Project Engineer

APPROVED: [illegible]

ROBERT D. KRUTZ, Colonel, USAF
Chief, Information Sciences Division

FOR THE COMMANDER: [illegible]

JOHN P. HUSS
Acting Chief, Plans Office

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19 **REPORT DOCUMENTATION PAGE**

READ INSTRUCTIONS BEFORE COMPLETING FORM

18 1. REPORT NUMBER RADC-TR-77-162 ~~Volume~~ 1 (of two)	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
6 4. ~~TITLE (and Subtitle)~~ IMPACT OF STRUCTURED PROGRAMMING STANDARDS ON SMALL GOVERNMENT CONTRACTORS. Volume I.		9 5. ~~TYPE OF REPORT & PERIOD COVERED~~ Final Technical Report, Aug 76 - Feb 77, 6. ~~PERFORMING ORG.~~ REPORT NUMBER N/A
10 7. AUTHOR(s) ~~Dr.~~ Wolfgang W./Gaertner		8. CONTRACT OR GRANT NUMBER(s) 15 F30602-76-C-0390 new
9. PERFORMING ORGANIZATION NAME AND ADDRESS W. W. Gaertner Research Inc 205 Saddle Hill Road Stamford CT 06903		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 16 62702F 17 14 5581/1411
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (ISIM) Griffiss AFB NY 13441		11 12. ~~REPORT DATE~~ May 1977 13. ~~NUMBER OF PAGES~~ 43
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same 12 50p.		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer:
Michael Landes (ISIM)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Structured Programming, Small Business, Small Contractor

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A study of the impact of the structured programming standards on small government contractors. How they can reasonably meet such requirements within the normal environment of the small contractor.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

388 544

LB

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(*When Data Entered*)

TABLE OF CONTENTS

TABLE OF CONTENTS

(continued)

EVALUATION

Probably the most clearcut result of the CCP-85 study on future needs of the Government, related to computer software, was the overwhelming requirement that large complex software systems be brought under control and be made amenable to thorough verification. As a result of the surfacing of this requirement, RADC initiated a many-faceted thrust aimed at enhancing software quality, reliability and maintainability. IS Division of RADC, under TPO V is engaged in an AFSC Software Engineering & Management Plan which contains focuses on four areas, Requirements Analysis (1.0), Management (2.0), Disciplined Programming Environment (3.0) and Quality Control (4.0). Under this plan, the RADC Structured Programming Series was produced containing the techniques and procedures of this discipline. Research at IS is also exploring language controls, testing, modeling, and other similar avenues which promise some solution to the problem of how to produce more reliable software. All products of this research are included as they mature in the RADC Software Development Specification CP 0787796100B, February 1977, which expedites their implementation on current software acquisitions.

Some concern was felt about the impact of such standards on the small government contractor. Would his normally constrained environment make compliance with the standards an undue hardship? If so, how should the standards be relaxed to accommodate his special needs? This report contains the results of a detailed study of the standards as they relate to this segment of industry. The study explains, in detail, how each of the Structured Programming requirements can best be met by a small government contractor. Distribution of such a technical report tells the small business contractors exactly how he can comply with each requirement mandated, without undue added expense, in the course of developing more reliable software on government contracts. The overall results of this study indicated that no relaxation of the standards was necessary so no degradation of quality would result from accepting proposals from small business.

[illegible]

MICHAEL LANDES
Project Engineer

A016791

I.1 INTRODUCTION

The concept and details of Top Down Structured Programming will not be redefined in this Report. Rather, the definitions and nomenclature of the Structured Programming Series (SPS), RADC-TR-74-300 (all volumes) will be adopted. This Introduction contains some general remarks on program development by Small Contractors and an outline of the approach taken in the analysis of the SPS.

The small R&D Contractor typically carries out a wide range of programming tasks in a variety of languages from microprogramming of microprocessors to higher-order-language programming of large-scale computers. In some cases, the program development is the main objective of the contract, and the final program is eventually run on the customer's machine. In other cases, program development is incidental to scientific research projects, but the programs developed may be deliverable to the customer. Or, programs are developed as part of a deliverable hardware system which is computer based or computer driven.

Program development may be carried out on the Small Contractor's own computer, on a large time-share system or on the customer's machine. In some cases where the computer on which the programs will ultimately run is not available for development, it may be emulated on a different computer. Such a case arises, e.g., with military avionics computers which are often not available for program development, but which can be emulated on large IBM 360's and 370's. In some cases programs are written without access to any computer, and the programs are delivered without having been tested. Verification and debugging is then carried out by the customer.

All contract types from CPFF to firm-fixed price, both prime and sub are utilized for program development. There is little doubt that the Small Contractor enjoys wide exposure to the full spectrum of computer-program development.

There prevails the notion in some quarters that the computer facilities available to the programmers in a given company are proportional to the size of the company: Where IBM might be using a System 370, a Small Contractor might use a pocket calculator. In fact, the inverse is often true. In the large company, the programmer is obligated to use the facilities and operating system of the available in-house computer; in the small company the programmer usually has ready access to large time-share systems which both from the standpoint of facilities and operating systems as well as support software are practically always superior to any given in-house computer system. As an example, W. W. Gaertner Research, Inc. routinely uses the NCSS IBM 370, model 168 and Amdahl 470, with 1 Mbyte core (not shared with any other customer) and virtually unlimited disk and tape storage. The Executive Language, the Edit facilities, the prepackaged routines and the system support software are superior to any available in-house at any large company. The response time of the system is from instantaneous to a few seconds, five days a week from 8 A.M. to 11 P.M., on Saturdays from 8 A.M. to 6 P.M. Hardware failures are infrequent, and down times typically only last a few minutes.

By comparison, W. W. Gaertner Research, Inc. is also currently carrying out a major project using another large customer-owned computer to which W. W. Gaertner Research, Inc. has a private-line connection. This system is significantly inferior in every aspect to the above time-share system, so much so that in several instances a few weeks of machine-language programming was necessary to achieve convenience features which are automatically available on the large time-share system. While, in principle, any company could use the large time-share computers that are available nationwide, company policy usually dictates the use of the available, and frequently inferior, in-house computer to guarantee the amortization of the procured hardware, and to keep the in-house computer staff busy.

In-house, W. W. Gaertner Research, Inc. maintains a PDP-11/40 minicomputer with disk and tape, as well as a NOVA minicomputer and several smaller machines. These are directly available to the programmers for interactive program development. Industry-compatible tape drives are used, with sufficient support routines to allow transfer of programs between the minicomputers and both IBM and CDC machines.

Program and data preparation are carried out on IBM Mag Card Selectrics, one of which has the Communicating Feature which allows it to directly access computers by simulating an IBM 2741 terminal. Thus data and programs are recorded magnetically, using routine secretarial help. They can be proofread and corrected, off-line, using standard office typewriters, and not requiring the tying up of computer terminals.

Punched cards are prepared only upon customer request, but are not involved in any program development.

Thus, while the facilities and modus operandi of W. W. Gaertner Research, Inc. may not be absolutely typical of the small R&D Contractor, the facilities are certainly affordable by any small company, and the lack of such facilities should therefore not be presumed a priori.

The large-company environment actually often requires program development in a card-based Batch Mode. The inefficiencies of this approach are staggering. In fact, if the Government is serious about reducing software-development costs, program development in batch mode should be outlawed before Structured Programming is strictly enforced. Chapter 5 of the SPS Volume VI shows some of the painful procedures necessary when simple editing is carried out in the card-based batch mode. According to SPS, Volume IX, page 3-19, the turnaround times for the various processes run from 21 minutes to 62 hours, with averages running approximately 4 hours. In the on-line mode these turnaround times would be seconds to minutes. The inefficiencies created by the programmer not having rapid feedback to his latest program changes, have a drastic effect on programming costs.

Since the Small Contractor practically never works in card-based batch mode , it will not be discussed in detail in this Report.

There are other differences between the large and the small companies. The small company has, of course, fewer people and usually smaller contracts. The smaller company has fewer but more senior programmers, typically with a wide exposure to different languages and machines. Most of them are qualified to carry an entire project by themselves. They work closely with the engineers and scientists and are usually familiar with the technical aspects of the project, rather than only its programming requirements. Top Down Programming has therefore usually been enthusiastically practiced, even before the technique got its present name. One result is that in the large company the average output per programmer is 5 to 10 lines per day (confirmed by SPS), whereas in the small company the output is typically 20 to 50 lines per day. Thus, for a given job, the small company would use smaller teams of programmers to accomplish the same job. This is further reflected in the fact that the cost per source-code line in the large company is typically $10 - $25 (see SPS, volume IX) whereas the small company costs start as low as $2 - $3 per line. These figures can be readily verified from past contractual records. All these figures are based on conventional program lines, rather than structured program lines which are shorter.

The above discussion defines the background from which the impact of the proposed Structured Programming Standards on the Small Contractor has been assessed. Top Down Programming will present no problem because it is generally practiced, and even if not, its implementation is not costly. The expense of conforming to the proposed Structured Programming Standards will depend on the rigidity with which details of procedures and reporting formats are enforced, and what programming and instructional aids will be made available to the Small Contractor to minimize the cost of the initial adaptation to the new standards.

W. W. Gaertner Research, Inc. has analyzed the entire Structured Programming Series, line by line, and prepared the following report which assesses its impact. The items in the proposed standards generally fall into the following categories:

a) Requirements which are either presently being met or can be readily implemented;

b) Requirements which can be satisfied with minor hardware and/or software investments, of a magnitude which could be absorbed by the first software contract requiring full compliance with the new Structured Programming Standards;

c) Requirements which would generate a software-development investment of a magnitude which could not be absorbed in a single, or a small number, of small software contracts and which would require Government-supplied programming and management aids.

Detailed comments in all these areas are contained in the following discussion of all volumes of the SPS.

This Final Report has been prepared in two Parts, I and II. Part I contains a summary of all significant comments, examples and recommendations. Part II contains all the detailed comments directly indexed to each paragraph of the Structured Programming Series (SPS) and also contains the more detailed examples. The numbering of the Chapters and Sections in Parts I and II has been chosen such that Chapters and Sections of identical numbers refer to the same material in the Structured Programming Series (SPS). A Chapter with a given number in Part II therefore contains the expanded discussion of the material summarized in the corresponding section in Part I.

Considering that the Structured Programming Series (SPS) consists of hundreds of pages of proposed standards it has obviously been impossible to copy into this report any material which is being analyzed. To understand this report, therefore, the reader must have a copy of the Structured Programming Series available to be able to read the material which is being assessed in the following chapters.

Also, it is recommended that this report be read in the order in which it is presented since there is considerable repetition and overlap between the various volumes of the SPS and pertinent comments are not necessarily repeated wherever the same subject is mentioned in the original volumes.

I.2 REFERENCES

1. Structured Programming Series, RADC TR-74-300, Volume I - Programming Language Standards, (A016771).

2. Structured Programming Series, RADC TR-74-300, Volume II - Precompiler Specifications, (A018046).

3. Structured Programming Series, RADC TR-74-300, Volume III - Precompiler Program Documentation, (A013255).

4. Structured Programming Series, RADC TR-74-300, Volume IV - Data Structuring Study, (A015794).

5. Structured Programming Series, RADC TR-74-300, Volume V - Programming Support Library Functional Requirements, (A003339).

6. Structured Programming Series, RADC TR-74-300, Volume VI - Programming Support Library Program Specifications, (A007796).

7. Structured Programming Series, RADC TR-74-300, Volume VII - Documentation Standards, (ADA008639). Volume VII - Addendum, (A016414).

8. Structured Programming Series, RADC TR-74-300, Volume VIII - Program Design Study, (A016415).

9. Structured Programming Series, RADC TR-74-300, Volume IX - Management Data Collection and Reporting, (A008640).

10. Structured Programming Series, RADC TR-74-300, Volume X - Chief Programmer Team Operations Description, (A008861).

11. Structured Programming Series, RADC TR-74-300, Volume XI - Estimating Software Project Resource Requirements, (A016416).

12. Structured Programming Series, RADC TR-74-300, Volume XII - Training Materials, (ADA026947).

13. Structured Programming Series, RADC TR-74-300, Volume XIII - Final Report, (A020858).

14. Structured Programming Series, RADC TR-74-300, Volume XIV - Software Tool Impact, (A015795).

15. Structured Programming Series, RADC TR-74-300, Volume XV - Validation and Verification Study, (A016668).

16. VP/CSS Reference Manual, Form 106, National CSS, Inc.

17. VP/CSS Edit Command, National CSS, Inc., March, 1975

18. Scope Reference Manual, Models 72, 73, 74, Version 3.4, 6000 Version 3.4, Control Data Corporation, 1974

19. Intercomm Reference Manual, Models 72, 73, 74, Version 4, 6000 Version 4, Control Data Corporation, 1972

20. RT-11 System Reference Manual, Digital Equipment Corp., June 1975, revision

21. Structured Programming Translations, RADC-TR-76-253, Final Technical Report, August 1976, Vol I, (A031457), Vol II, (A031458), Vol III, (A031459), Vol IV, (A031460) and Vol V, (A031461).

I.3 Impact of the Guidelines and Standards of the Structured Programming Series on Smaller Government Contractors

I.3.1 General Comments on Chapter I.3

This chapter is a summary of Chapter II.3 to which the reader is referred for detailed comments and recommendations, as well as examples which illustrate the intent of the Structured Programming Series and which also show that it is often very straightforward to implement the requirements of the Structured Programming Series on all three computing media typically used by the Smaller Government Contractor, namely the in-house minicomputer, the time-share service, and the large Government-owned computer.

The review and analysis is summarized in this Chapter I.3, followed by all pertinent recommendations. The latter deal primarily with programming aids which might be provided by the Government to the Smaller Contractor (and which would generally also be useful to the Larger Contractor), and to a lesser extent with suggested changes to the Structured Programming Series. The purpose of these suggested changes is usually not to make compliance with the Structured Programming Series easier for the Smaller Contractor, but to make the standards more logical and consistent, and to orient them primarily towards interactive on-line program development, as opposed to card-oriented batch program development.

A management-type overview of the problems associated with the use of Structured Programming by the Smaller Government Contractor can be gained from Volume I of this Report alone. However, a full appreciation of the subject will require a study of both Volume II of this Report and of the entire 15 volumes of the Structured Programming Series.

I.3.2 Programming Support Library

I.3.2.1 Review and Analysis

The guidelines for the implementation of a Programming Support Library (PSL) are contained primarily in Volumes V and VI of the SPS. More specifically, Volume V deals with the Functional Specifications and Volume VI with the Program Specifications for a PSL.

Two categories of specifications are established, namely "Basic" and "Full", which are defined as follows:

Basic - These requirements are the minimum needed to provide a basic library capability. A PSL implementation which satisfies this set of requirements is appropriately used by programming groups involved in the development of independent programs rather than program systems. Usually this will be used in computer system environments which cannot tolerate the additional computing resources (time and space) required by a PSL which satisfies the Full set of requirements.

Full - A PSL implementation which satisfies this set of requirements is appropriately used for support of system development and/or larger programming groups developing application programs.

A summary of the Functional Requirements according to Volume V of the SPS is shown in the following table:

SOURCE DATA MAINTENANCE

Basic Requirements

Data File Storage

Program Source Code

Program Object Modules

Program Load Modules

Job Control Data

Test Data

Program Design Language (PDL) Statements

- Textual Data
- Other Data

Data Access

Library Backup Capability

- Backup Storage independent of the master file
- Regeneration from Backup Storage

Data Maintenance

- Add one or more units of data to a file
- Delete one or more units of data from a file
- Replace one or more units of data with new units of data
- Make the following changes to one or more units of source data
    - Add one or more lines of data
    - Delete one or more lines of data
    - Replace one or more lines of data
- Organize data for easy reference
    - Automatic generation of line sequence numbers
    - Regeneration of line sequence numbers
    - Capability to add lines without regeneration of line sequence numbers
- Copy one or more units of data from one file to another
    - Within a single library
    - Between libraries

Full Requirements

- Data File Storage
  - Compression of Data
  - Restoration of Data
- Data Maintenance
  - Make the following changes to a unit of source data
    - Modify a line of data
    - Scan for and replace a specific string of data
    - Insert a string of data
  - Make temporary changes to unit of data
  - Automatically generate program stops
  - Merge to PSL data files from two different PSL libraries

OUTPUT PROCESSING

- Basic Requirements
  - Library Control Listings
    - Information related to Physical Storage
    - Directory Listing
  - Source Data Listings
    - Listings of selected units of data
    - Listings of all data in a single library data file
  - Control Data
- Full Requirements
  - Magnetic Tape Output
    - Copy one or more specific units of data onto magnetic tape

Copy one or more files of data onto magnetic tape

Punched Card Output

Punch contents of one or more units of data

Punch contents of one or more files of data

Management Control Listings

Programmer Directory and Unit Listings

Automatic Indentation of Source Code Listings

Data Scanning Capability

PROGRAMMING LANGUAGE SUPPORT

Basic Requirements

Compiler Interface

Syntax Checking of Source Code

Compiling of Source Code

Load Module Generation

Full Requirements

Top Down Structured Programming Support

Flagging of all explicit branches

Flagging program language source code units that exceed a maximum size defined by the user

Flagging any lines of program source code that contain more than one source statement

LIBRARY SYSTEM MAINTENANCE

Basic Requirements

Library Installation Support

Library Maintenance Support

- Library Installation Support
- Library Maintenance Support
  - Allocation of new library data storage and directory space
  - Reallocation of existing library data storage and directory space
  - Compression of data
- Library Termination Support

Full Requirements

- Library Installation Support
  - Definition of the hardware configuration and installation time to meet the user's requirements
  - System generation facility that allows the user to generate a PSL that contains only the major functional capabilities

DATA SECURITY

Basic Requirements

Full Requirements

- Data Integrity
- Data Protection
  - Access restrictions for libraries within a Project and for data files within a library
  - Access restrictions to data units stored in a library file
    - Restrict access for either output or update
    - Allow free access for output but restrict update
    - Allow free access for output or update

Printing of classified titles

MANAGEMENT DATA COLLECTION AND REPORTING

Basic Requirements

Full Requirements

Collecting

Counting input source code etc.

Start date, end date, etc.

Number of compilations, number of assemblies, lines of code from another source, etc.

Allow addition of routines for gathering of actual data

Updating

Allocate Management Statistical Data Storage Areas

Edit Management Statistical Data

Add Management Statistical Data

Delete Management Statistical Data

Replace Management Statistical Data

Computer Job Turnaround Time

Termination of data collection for specific types of data

Termination of data collection and archival functions

Accumulating

Summarizing Management Statistical Data

Processing Management Statistical Data

Archiving

Store all data items

Backup of Management Statistical Data

Reporting

Produce Program Module Statistics Reports

Produce Detailed and Summary Computer Utilization Reports

Produce Program Maintenance Statistics Reports

Produce Program Structure Reports

Produce Historical Reports

Produce Combination Reports

Produce reports that include cycled data

Allow user to add routines to the PSL

Produce reports on exception basis when significant variances exist between actual and planned data

DOCUMENTATION SUPPORT

Basic Requirements

Full Requirements

Print in a user-specified sequence one or more units of program stored in the PSL

Format an output listing with the following user-specified information:

Beginning of page

Header information on top of page

Spacing between units of data

Spacing between lines of output

Number of lines on page

Automatic page numbering

- Printing of Title Page
  - Document Title
  - Document Date
  - Author's Name(s)
  - Organization and Address
- Generate a magnetic tape in print image form

GENERAL REQUIREMENTS

- Basic Requirements
- Full Requirements
  - Concatenated Data Files
  - Subroutine Support

ON-LINE TERMINAL IMPLEMENTATION REQUIREMENTS

- Support of terminals via communication links or directly via cables
- Support of both display and typewriter terminals
- Remote job entry capability
- Terminal-to-terminal communication capability
- Collection of statistics related to the use of each on-line terminal
- Facility to recover from system failure
- Restrict access to specific PSL data files to specific on-line terminals
- User identification and authentication for access
- On-line user support facility
- Paging capability to browse or scan through data
  - Display next page of data

Display previous page of data

Display last page of data

Roll display forward

Roll display backward

Chapter II.3.2 of this report discusses each of these requirements in detail and provides examples for their implementation on in-house minicomputers (PDP-11), commercial time-share services (IBM 370, AMDAHL 470) and a typical Government computer installation (CDC 6600), all of which might be used by the Small Contractor in his program development work.

Compliance with many of these requirements is straightforward on all three types of computer systems, in which case the imposition of the Structured Programming standard will present no problem to the Small Contractor. However, the implementation of a number of requirements will entail the generation of additional support software of varying complexity.

The Basic Requirements which cannot be readily implemented with existing software on an in-house minicomputer are

the automatic generation of line sequence numbers,

the automatic regeneration of line-sequence numbers,

the listing of specific lines of data in a file,

the automatic addition of internal library and unit names, and of date and time to all printed listings.

The Basic Requirement to be able to add lines of data to existing units without regeneration of line sequence numbers is not readily possible on the CDC 6600 which has been chosen as the typical Government computer.

Syntax checking and Library System Maintenance are two *other* Basic Requirements which may largely be satisfied by existing software on all three computer systems typically used by the Small Contractor, but it is fairly certain that additional software will be needed in both cases to make compliance with the requirement more complete and convenient.

Full Requirements, not presently available on any existing software system are

the automatic generation of program stubs,

the automatic generation of programmer directory and unit listings,

the automatic indentation of source-code listings,

the automatic flagging of all explicit branches,

the automatic flagging of program language source code units that exceed a maximum size defined by the user,

the automatic flagging of any lines of program source code that contain more than one source statement,

all Full Requirements for Collecting, Accumulating and Reporting (but not of Updating and Archiving) of Management Data (SPS Vol. V, Chapter 3.6.2).

Not presently available on the typical in-house minicomputer are the following Full Requirements:

Complete compression and restoration of data through elimination of leading, trailing and intervening blanks,

Access restrictions,

Generation of magnetic tape in print-image form,

Generation of use statistics.

To comply with the Full Requirements, the CDC 6600 system would also need routines for

data compression and restoration,

insertion of strings of data into specific positions of every line in a unit of data.

Furthermore, automatic page numbering, not starting with page number 1, as well as the automatic generation of title pages is also not readily available for the minicomputer and the CDC 6600.

Volume VI of the Structured Programming Series provides the specifications for all the support programs considered necessary to implement the functional requirements of Volume V. These are

Main PSL Programs/Subprograms

Program	Function
LSM	LIBRARY SYSTEM MAINTENANCE
INSTALL	Initialize a PSL installation
INITIAL	Initialize a PSL project
ALLOCATE	Allocate, reallocate, or compress PSL Library files; initiate/terminate management data collection
TERMINATE	Terminate PSL projects or delete PSL libraries and files
SDM	SOURCE DATA MAINTENANCE
UPDATE	Update PSL library files
MERGE	Merge PSL library files
BACKUP	Backup file/library/project
RESTORE	Restore unit/file/library/project
OUTPUT	OUTPUT PROCESSING
INDEX	Print file directory index listings
SOURCE	Print listings of source data stored in the library
AUTHOR	Print index or listings of units developed by a programmer
CSCAN	Scan for specific character strings
MDCR	MANAGEMENT DATA COLLECTION AND REPORTING
MDFILE	Initialize/terminate manual management data collection for a project
MDUPDATE	Update management data files
MDREPORT	Establish automatic management date reporting
MDPRINT	Print management data reports
MDHIST	Print historical management data
MDXCHECK	Establish automatic exception checking reporting
MDUSER	Provide user routine interface

Program	Function
PLP	PROGRAM LANGUAGE PROCESSING
COMPILE	Provide precompilation/compilation interface
LINK	Provide program linkage interface
EXECUTE	Provide execution interface
MISC	MISCELLANEOUS TERMINAL SUPPORT
TTABLE	Initialize and maintain authorized terminal/project lists
MESSAGE	Terminal-to-terminal message formatting
TSR	Terminal statistics reporting
<u>PSL Support Software</u>	
Operating System	Controls the execution of PSL programs and processes job control statements stored in the library
Allocation Program	Controls the allocation of storage space on direct access storage devices
Data Access Program(s)	Supports the reading and writing of library data files on either a direct access storage device or magnetic tape
Assembler	Assembles source statements
Precompiler	Accepts structured source code from the SOURCE file and prepares input for the appropriate compiler
Program Source Language Compilers	Compiles the program source language statements
Link Program	Combines one or more object and/or load modules into an executable load module
Loader	Transfers load module into core storage for execution.

In the following we shall identify those programs whose functions are largely already available in existing software on the three types of computer systems typically used by the Small Contractor, and those programs which will have to be developed and supplied to the Small (and probably Large) Contractor:

Library System Maintenance

The functions of the following programs are already largely available from existing software:

INSTALL

INITIAL

TERMINATE

NTABLE.

From the ALLOCATE program the only functions which are not available from existing software are:

Structured Programming checking,

Management Data Collection.

Source Data Maintenance

It is believed that the functions of the

UPDATE (= crude editor),

MERGE,

BACKUP,

RESTORE

programs can already be provided by existing software.

Output Processing

The functions of the

INDEX,

CSCAN

programs are available from existing software, but the

AUTHOR

program should be provided in full, as well as certain features of the

SOURCE

program, such as

Automatic Indentation.

<u>Management Data Collection and Recording</u>

It is believed that the features of the

MDUPDATE,

MDUSER

programs are already available, but that programs with the capabilities of

MDFILE,

MDREPORT,

MDPRINT,

MDHIST,

MDXCHECK

will have to be provided.

<u>Programming Language Processing</u>

The programs

LINK,

EXECUTE

are probably not required, but there is a definite need for a

PRECOMPILER, and

the "COMPILE" program which provides the precompiler/compiler interface.

Chapters I.3.5 and II.3.5 contain a more detailed discussion of the subject.

Documentation Processing

No new program appears necessary to support this requirement.

Additional Terminal Functions

The TTABLE message programs are probably already implemented by existing software, but a

Terminal Statistics Reporting (TSR)

program for in-house minicomputers may have to be provided.

I.3.2.2 Recommendations

It is recommended that the specifications of Volumes V and VI of the Structured Programming Series be revised to make them clearer, especially by providing specific examples, and to also make them more compatible with a large amount of existing software (operating systems, utility routines, editors, executive languages, etc.) through which many if not most of the Functional Specifications can currently be implemented.

Also, the Program Specifications of Volume VI are much more restrictive than the Functional Specifications of Volume V. In fact, they must be considered too narrow since many other programs than the ones listed in Volume VI already exist which satisfy the intent of the Functional Requirements.

It is also recommended not to look at the PSL Programs as a separate entity, but to allow the maximum utilization of existing software if it satisfies the Functional Requirements.

The nomenclature of Volume VI should not be frozen since different software systems use different terms for the same concepts and it does not appear advisable to introduce terms which conflict with the nomenclature of the operating system and utility routines of a given computer system (as an example, "Text" may either denote alphanumeric textual information, or an object module, depending on the conventions employed in a given computer system). Again, it would be highly desirable to generate specific examples of each SP concept on all major computer systems. It is also recommended that in all future specifications the Terminal Environment be heavily emphasized over the Batch Environment.

Since many of the Full Requirements can be readily met by the Small Contractor (and therefore presumably by any Contractor), it appears that many of these requirements should be considered Basic. As a start, all the Full Requirements with which the Small Contractor can readily comply, according to Volume II of this Report, might be shifted into the Basic category.

Data Compression/Restoration should become a Basic Requirement, since Structured Programming lines are generally short and uncompressed storage is very wasteful.

The requirement to provide punched-card output might be removed, since the punched card must be considered an obsolescent medium.

The most important recommendation concerning the Programming Support Library aspect of the SPS is that the Government provide support-program packages for all the functions which cannot currently be readily implemented by existing software. These have been identified in Chapter I.3.2.1 above.

Two types of packages might be provided, one for Basic Requirements and another for Full Requirements, after these have been redefined according to the preceding recommendations.

The preferred approach would be to supply these programs in HOL to be compiled by the user. This will not always be possible and assembly or machine language modules for specific computer systems may also have to be provided.

In addition to the programs which should be supplied by the Government, it would also be very useful to provide instructions on how to utilize the Executive Language and JCL facilities of existing systems, and possibly to supply guidelines on how to modify existing operating systems and utility routines to create the capabilities needed to satisfy the Functional Requirements of Volume V of the SPS.

In all cases, the heavy use of detailed examples is strongly encouraged.

I.3.3 Applicability of Structured Programming Standards and Guidelines

I.3.3.1 Review and Analysis

The broad intent of the Structured Programming Series is contained in SPS Vol. I, Programming Language Standards and SPS Vol. IV, Data Structuring Study. The detailed review of these volumes, given in Chapter II.3.3 of this Report indicates that the entire concept of TDSP is applicable to the work of the Small Contractor. In fact, many of the suggestions and requirements of the SPS are already routinely being fulfilled by the Small Contractor, as is proven by the many examples given in Volume II of this report.

The comments contained in Chapter II.3.3 of this Report pertain primarily to the logic aspects of the requirements of the SPS and take no issue with the applicablity of TDSP to the tasks normally performed by the Small Contractor. They are, therefore, not summarized in this Chapter I.

In general, the Small Contractor could expect to realize a competitive advantage from the use of Structured Programming. While it is well known that the cost per coded line is considerably less when programs are prepared by Small Contractors as opposed to Large Companies, the customer is often concerned about the ability to maintain programs should the Small Contractor go out of business. The customer may therefore contract with the Large Company to be assured of continued program maintenance. On the other hand, if the Small Contractor used widely accepted standards for Structured Programming, and delivered his product in such well documented form that the customer could be assured of program maintenance by an alternate source, the customer may be more willing to take advantage of the cost savings provided by the Small Contractor in original program development, since the danger of not being able to maintain the program later would be greatly diminished. By making programs easier to write, the need for large programming staffs should also be reduced, and it should be possible for the Small Contractor to handle bigger projects. This, in turn, will increase competition among competent programming sources, and the Government should benefit from this increased competition. For this reason, it is important that the transition to full compliance with the Structured Programming Standards should be made as easy as possible for both the Small and the Large Contractor.

I.3.3.2 Recommendations

Since all aspects of the proposed TDSP standards will be applicable to the work carried out by the Small Contractor it is recommended that the Government take steps to ensure that it is easy for the Small Contractor to comply. This can be accomplished by two initiatives:

a) Provide clear unambiguous TDSP standards, fully compatible with the operating systems, utility routines, editors, etc. of modern computer systems, illustrated by a large number of meaningful examples,

b) Supply aids, primarily in the form of support programs, wherever the cost of implementing the entire TDSP standard cannot be amortized over a single programming contract with a level of effort of approximately 1 to 2 man years. These suggested aids are identified throughout this Report.

I.3.4 Chief Programmer Team Operations

I.3.4.1 Review and Analysis

The subject of the Chief Programmer Team Operations is discussed primarily in SPS Volume X. According to the Conclusions and Recommendations of Volume X, this concept has never been fully tested, nor is its full implementation immediately recommended.

Under a smaller software contract, the teams are small and the full administrative structure recommended in Volume X would often not be applicable. In fact, several of the functions of Chief Programmer, Backup Programmer, Secretary/ Librarian and the Support Members may all only be part time on a given project. In some cases, the Chief Programmer, may, in fact, not be a full-time programmer, but a scientist or engineer with programming background. Such a person has the advantage, well recognized on page 2-2 of Volume X, that "comprehension of a fairly complex design and implementation can reside in one person. Significant advantages are reported as a result of the simplifications and unified structure which can be achieved by one mind in designing the entire system." The person also has the advantage that he or she fully understands the technical aspects of the program's purpose. Such scientists/ programmers generally adhere to the principles of Top-Down Programming and they are readily capable of generating meaningful tests for the verification and validation of the program. On the other hand, they must be taught the principles of Structured Programming to assure adherence to the proposed standards. Such a scientific/engineering-based Chief Programmer has the additional advantage of being able to judge the degree of completion of the project, not just by counting the number of coded lines, but by assessing the percentage of the overall technical task which has been completed. In fact, since the part-time programmer part-time engineer/scientist is a fairly common occurrence with the Small Contractor, the organization of an efficient and reliable programming effort based on this type of personnel should be described in greater detail in a future revision of the Structured Programming Series. The other type of Chief Programmer, typically found in the Small Company, is a senior programmer with management capabilities. The Small Contractor often has a core of Chief Programmers, each with a solid software background in various languages from machine code to HOLs. In addition, these individuals usually possess broad knowledge of various operating systems and hardware. Since many projects carried out by the Small Contractor support only a single programmer, the majority of the programming personnel of the Small Contractor must be capable of exercising the duties and responsibilities of the Chief Programmer.

Projects requiring the services of several programmers may be organized such that one of the programmers assumes the role of the Chief Programmer while other, equally qualified programmers serve as Backup Programmers on a specific project. Their roles may well be reversed on another project depending on their familiarity with the subject matter of the program's purpose.

The Secretary/Librarian function is presently largely in existence, certainly as far as program and data inputting is concerned. It has been readily possible to train intelligent secretaries with some computer-terminal experience to satisfy all the functions of the Secretary/Librarian.

The Small Contractor will generally not employ a significant number of Backup Programmers and Support Members, since most such companies are staffed to carry out primarily one to three-man-year programming efforts requiring primarily the services of senior, rather than support personnel.

I.3.4.2 Recommendations

It is recommended that the concept of the Chief Programmer Team Organization be tested in a variety of environments, some including the Small Contractor and that the guidelines be revised, and very concrete examples be given.

The case where the Chief Programmer is actually a scientist or engineer in charge of the entire project which may include other investigations than program development alone, should be given special consideration. It is of considerable significance to the Small Contractor, but also occurs frequently in Large Companies where computer-program development is part of an overall scientific investigation.

There is no particular time pressure concerning the development of standards for the Chief Programmer Team, since all other aspects of a proposed TDSP standard could be implemented without relying on the Chief Programmer Team.

I.3.5 Higher-Order Languages

I.3.5.1 Review and Analysis

The mix of languages used depends strongly on the Small Contractor's main business activity. If it lies strictly in the scientific, business, data-base or management system areas, he will almost exclusively use the higher-order languages and the proposed TDSP standards will be directly applicable.

If the company works on the development of actual hardware, especially custom systems which require highest possible throughput, or if he is developing or modifying operating systems, he will work more in the lower-order languages, all the way down to microprogramming of microprocessors, microcontrollers and the ALU nucleus of minicomputers. The same applies to the programming of microprocessor arrays and other novel parallel/associative computer structures such as Goodyear's STARAN, the ILLIAC IV or W. W. Gaertner Research, Inc.'s G-471. For work in the higher-order languages, precompilers or translators such as those described in SPS Volumes II and III, as well as the Structured Programming Translator for FORTRAN (reference 21, see Chapter I.2 of this Report) will be of particular value.

The applicability of Structured Programming Standards to pure machine-language programming has been questioned since there are indications that Structured Programming may reduce throughput below that achievable by hand coding without regard to any programming standards. The execution delays introduced by adherence to Structured Programming Standards may be intolerable in certain real-time applications. This subject will have to be resolved in future R&D contracts. However, the outcome will affect the Small Contractor in the same way as the Large Contractor.

Another type of program development could also create a potential conflict with the new SP standards: For smaller minicomputer work, and for microprocessors, terminals, communication equipment, etc., the available hardware would be below the minimum necessary for effective utilization of the SP tools. In such cases for SP to be applied, the smaller processor would have to be emulated on a larger computer system which can provide the support software for the SP Standards. Fortunately, links between the smaller processors and larger disk-operating minicomputers can be established fairly readily and a mechanism could probably be developed to apply Structured Programming to microprocessor and other small computer work.

I.3.5.2 Recommendations

The most important recommendation on the subject of Higher-Order Languages is for the Government to perfect precompilers and translators and to fully test them on the various major computer systems likely to be used in Government program development by Small or Large Contractors alike. These fully debugged and documented precompilers could then be made available to any Government contractor. This approach is preferable to letting each Contractor develop his own precompiler, since the estimated cost, according to SPS Volume XIII might be $20,000 per language with additional costs for conversion to computer systems other than the one on which it was originally developed. On the other hand, a Government-supplied precompiler or translator should be operational on an in-house system within one man week.

Another desirable project would be for the Government to develop techniques for using larger popular minicomputers like the PDP-11/40 or bigger to serve as the host for Structured Programming development work on microprocessors, intelligence terminals, programmable communication equipment, etc. With the proliferation of microprocessors into many types of military equipment, and the notorious difficulties in documenting and maintaining microprocessor software, this project might be of major benefit to the development of all micro-processor based military equipment.

Another investigation might be concerned with the potential of applying SP standards to the programming of automatic and built-in test equipment.

A further desirable study concerns the resolution of the suspected conflict between SP standards and highest-throughput real-time machine-language programs.

Most of these recommendations affect the Small Contractor in the same way as the Large Contractor, except for the precompiler and translator development whose cost might be fairly readily absorbed by a Large Company, whereas the Small Contractor would rely on Government-supplied support software. The advantage to the Government in developing and supplying uniform precompilers and translators is that it could control the quality of these support programs and thereby insure a higher level of compliance with the new SP standard than if a wide variety of uncontrolled precompilers and translators were developed by the different Contractors.

I.3.6 Subcontracting Provisions Imposed on the Small Contractor

As discussed in greater detail in Chapter II.3.6, there are no significant differences between a prime contract or a subcontract for program development. The Small Company would follow very similar procedures in both cases.

The only unusual case would arise if the Large Prime Contractor required the Small Contractor to generate all programs and PSL output in formats which are fully compatible with the Large Prime Contractor's own PSL. In that case, the Large Contractor may have to supply specific programming aids to the Small Contractor, or make his own facilities available for program development.

I.3.7 Software Development Tools

The subject of Software Development Tools is addressed in SPS Volumes I, Programming Language Standards; Volume VIII, Program Design Study and Volume XII, Training Materials. In addition, the precompilers and Structured Programming Translators discussed in Chapter I.3.5 of this Report must also be considered important Software Development Tools. Chapter 7 of SPS Volume VIII recommends specifically the formal introduction of HIPO, Structured Design and a PDL. No truly convincing arguments are being proferred for the use of these tools. However, whatever the ultimate decision with respect to Software Development Tools, there will be no difference in their use between the Large or Small Contractor, except that the Large Contractor might provide in-house training seminars for their programming staff, an approach which may not be financially feasible for the small company.

I.3.7.1 Recommendations

It is recommended that the development of programming aids be pursued further, in addition to the precompilers and translators which have been discussed earlier in Chapter I.3.5 of this Report. After the value of a specific Software Development Tool has been established, the Government might consider the preparation of training aids such as manuals and even seminars to familiarize Small Contractors with the outcome of the development programs for software development tools. Of particular help would be the publication of detailed case histories on program developments in the major languages which were carried out in full compliance with the Structured Programming Standards and heavily supported by any new Software Development Tools. This would save the Small Contractor potentially costly arguments over the interpretation of the standards and would also rapidly familiarize him with any development tools which have been found valuable and whose use might eventually be considered mandatory.

The development and promulgation of specific Software Development tools does not appear extremely urgent at this time, since their absence would not prevent the introduction and use of a Structured Programming Standard.

As the example given in Chapter II.3.7 of this Report shows, the Small Contractor would generally already employ systematic procedures to develop TDSP programs which would be adequate for the initial implementation of an SP standard without additional Software Development Tools.

I.4 Automated Management Tools

I.4.1 Management Reporting Tools - General Financial and Management Information System

I.4.1.1 Review and Analysis

The subject of Management Reporting Tools is discussed primarily in SPS Volume IX, Management Data Collection and Reporting, and Volume XI, Estimating Software Project Resource Requirements. In this Chapter I.4.1 we discuss primarily the financial aspects of Management Reporting, whereas Chapter I.4.2 deals with techniques for monitoring the Technical Progress of a project.

Obviously, it is extremely important to properly connect the financial and the technical aspects of program development.

The value of accurate and timely financial reporting in the Small Company is primarily to monitor expenditures (and compare them with the percentage of project completion), rather than in accumulating historical information which eventually, on a statistical basis, could be used in future estimating of program development costs. Past experience is frequently not a very good guide for estimating programming costs, since both the characteristics of programs and of programmers change sufficiently rapidly for previous experience to be only marginally applicable. Also, total program-development costs are frequently dictated by a level of effort stipulated by the customer.

However, accurate financial monitoring of program-development costs is vital to the survival of the company and is usually carried out conscientiously. The categories in which costs must be accumulated are properly identified in SPS, Volume IX, page A-8, "Resource Cost Data Items" and page A-10, "Computer Utilization Data Items". Not specifically mentioned, but obviously of great importance are the overhead and G&A costs which must be added to the direct costs listed in the above reference, since the indirect costs are usually at least of equal magnitude to the direct costs.

The report of greatest interest to the Small Contractor is that portion of the Historical Report, SPS, Volume IX, page B-15 which deals with accumulated cost of the project. The preparation of this report is readily affordable by the Small Company and a typical procedure to obtain the necessary input data is as follows:

The first step is an unambiguous definition of milestones to be achieved, typically defined as completed (debugged and tested) program modules. A separate charge number is assigned to each of these program modules to allow separate accumulation of costs. All costs, including labor hours by employee number, material, subcontracts, travel, computer costs, consultants, and other expenses are accumulated on a daily basis, separated by charge number, and are fed into the computer together with any overhead and G&A costs incurred. At the end of each reporting period, typically once per week, a set of financial analysis programs is executed which produces for each charge account, each project number, overhead and G&A accounts, the direct labor, material, subcontract, travel, computer costs and other expense figures for the current reporting period and accumulatively for the current Fiscal Year, and accumulatively for the total duration of each active charge number. One of the unique features of the program developed by W. W. Gaertner Research, Inc., and not usually found in the prepackaged financial information systems of large companies, is the fact that the overhead and G&A for the current Fiscal Year is completely recalculated at each execution of the program, so that the resultant cost figures are not estimates based on projected overhead and G&A figures, but represent the actual cost. This is of considerable importance, since the after-tax profit on Government Contracts lies between 2 and 7% and an erroneous prediction of overhead and G&A rates by more than 7% can make the difference between showing a profit and incurring a loss. A side benefit of this automatic financial monitoring system is the ease with which all accounting information can be audited, which is frequently necessary on all Government contracts.

All costs are compared against budgets and any deviations are noted. The program-module cost summaries contain employee numbers, names and hours spent. There is also cross indexing showing for each employee the allocation of his time over the various active program modules. In-house computer usage is clocked and assigned to program-module numbers, time-share costs are monitored by CPU seconds and I/O cost units, and are also readily assigned to program modules.

Thus, a sophisticated financial management information system is readily affordable by the Small Contractor, and might even be simulated by manual techniques. It is, of course, of great importance to closely relate the accumulative costs to the technical progress, as discussed in Chapter I.4.2 below.

I.4.1.2 Recommendations

Since even very Small Government Contractors usually maintain a detailed financial reporting system, primarily to satisfy the requirements of the Defense Contract Audit Agency, it will generally not be necessary for the Government to provide additional financial monitoring tools. However, recommendations should be made to establish separate charge numbers for all milestones in the overall program development, and to compute accumulated costs sufficiently rapidly after their occurrence such that corrective action can be taken when deviations between actual and budgeted costs occur.

I.4.2 Collection of Program Development Data

I.4.2.1 Review and Analysis

The subject of the collection of program development data is addressed primarily in SPS Volume IX, Management Data Collection and Reporting, and Volume XI, Estimating Software Project Resource Requirements.

The main value of these techniques from the standpoint of the Small Contractor is to improve the monitoring of the Technical Progress under the project, and perhaps secondarily to develop better resource estimating techniques. Obviously, the recommendations of the SPS are not final, considering the amount of doubt expressed in the documents themselves. As an example, while the current length of the program, measured in the number of lines is, of course, readily available from any listing of disk files, a long program is not necessarily a superior program, but might only be the result of highly inefficient coding. The removal of source-code lines may be a more brilliant programming achievement than an addition. Under the Top-Down approach, the number of compiles and assemblies for a given period is no measure of programming quality either. While in the past one might have assumed that an excessive number of compiles indicate a large number of errors in the source code, under the Top-Down approach it may simply be that a program is being carefully tested step by step as it is being developed. Verifying the program in small incremental steps is probably the safest and least expensive way of assuring reliable software.

A more directly valuable method for measuring project progress which is probably applicable to a wide range of Small Contractors is as follows:

Milestones, expressed in terms of operational program units, are established at the beginning of the project and tied in with a dollar budget, usually broken down into programmer hours, computer costs and other direct and indirect costs as discussed in the preceding Chapter I.4.1. Under the TD approach, a version of the program is always operational and verified, so that it is relatively easy to determine whether a certain milestone has been reached. The number of lines of source code, the number of compilations and assemblies as well as test executions necessary to reach a given milestone is not nearly as important as the total cost incurred. In fact, a programmer has the option of using more computer time and fewer programmer hours if a particular mix produces the least cost.

From the standpoint of the Small Contractor a very large amount of the data suggested as management information in Volume IX of the SPS seems of little value since it seems intended more for the development of long-term statistical information than to monitor the status and progress of a given project.

I.4.2.2 Recommendations

It is recommended that the suggested Management Data Collection and Reporting requirements be drastically reduced to primarily supply information which measures the status and progress of the development of the various active program modules. This information should contain the planned start and end dates for the module development, the planned programmer hours and other resource costs and the programmer's current estimate of percentage of completion, estimated completion time and statement as to the adequacy of the remaining resources, all updated prior to log out on the latest development session of the particular program module.

As a measure of activity, rather than progress, the number of source code lines, compiles and assemblies might also be accumulated, primarily to serve as an indication to management that a given program module is actually under development. A listing of the dates and durations of terminal sessions concerned with a given program module might also be readily accumulated for review by management.

It is therefore specifically recommended that the minimum status and progress reporting items of true value be defined and their automatic collection be integrated into the PSL support programs which will ultimately be supplied by the Government to Small (and perhaps Large) Contractors.

I.4.3 Error Data Reporting

I.4.3.1 Review and Analysis

The subject of Error Data Reporting is imbedded in SPS Volume IX, Management Data Collection and Reporting. The errors of concern are

keying/clerical errors, i.e. the number of keying/clerical errors for a program that were corrected by source code update executions;

specification omission errors, i.e. the number of specification omission errors for a program that were corrected via source code update executions. These errors were made because of omissions in the program specification;

specification misstatement errors, i.e. the number of specification misstatement errors for a program that were corrected via source code update executions. These errors were made because of misstatements in the program specification;

implementation misinterpretation errors, i.e. the number of implementation misinterpretation errors for a program that were corrected via source code update executions. These errors were made because of misinterpretations of the program specification;

implementation incorrect errors, i.e. the number of incorrect implementation of functions described in the program specification that were corrected via source code update execution;

implementation omission errors, i.e. the number of non-implementation of functions described in the program specifications that were corrected via source code update executions.

In general, no formal Error Data Reporting Techniques seem to be followed by either the Large or the Small Contractor at the present time.

It is, of course, recognized that errors which are rapidly recognized and corrected in the interactive on-line environment do not anywhere near have the devastating effect they have in the batch environment with its excessive turnaround times. In the on-line terminal environment there is a definite trade-off between the cost of careful double checking of source code vs. processing it rapidly through a sophisticated syntax checker, compiler and actual test cases, thereby rapidly removing all residual errors.

Also, the value of detailed Error Data Reporting must still be proven. Presumably, it would be of great interest to management to determine exactly what type of errors are committed by which programmer or support personnel to accomplish objective performance evaluations and to institute training programs to remedy the causes for the most frequently committed errors. On the other hand, there is a very strong objection on the part of the programmers to have their work monitored in this detail. More importantly, frequent editing and changes of source code may be a valid and efficient program development technique. Especially in the on-line environment it is very feasible to rapidly compile and test program segments which deliberately have been kept simpler and/or different from the ultimately desired source code, simply to continuously verify all partial source code practically up to the last line coded. Under this procedure, frequent changes in source code would not be an indication that the particular programmer commits a large number of errors.

Assuming that the value of Error Data Reporting can be proven and the establishment of a standard is contemplated, the Small Contractor would not be significantly affected if the desired Error Data Reporting techniques were largely manual. Should automatic collection techniques be stipulated, perhaps along the lines described in the following paragraph, then the Government should probably provide the necessary support program to the Small Contractor rather than asking him to develop these techniques as overhead on relatively small programming jobs.

I.4.3.2 Recommendations

The first recommendation concerning Error Data Reporting is for the Government to select one or more Large Contractors and test the value of the concept.

Assuming that Error Data Reporting is considered essential, an automatic collection technique could be developed as follows: The Editor of the computer system could be modified such that any changes, additions and deletions of source code require the insertion of a comment before such a modification step is executed. The comments, to be inserted by the programmer, would contain an explanation why the particular change is undertaken and would classify the cause of the change into one of the error categories listed in Chapter I.4.3.1 above, or otherwise explain the change as a normal programming step. The operating system would be further modified to allow full recording of each program-development terminal session, including the time records. These recorded terminal sessions would then be subjected to an analyzer program which could readily collect the type of source-code changes made by the programmer together with the comments as to the error causes necessitating the particular change. Since all changes would also be annotated with actual time, it would even be possible to determine how rapidly a given programmer modifies the source code. Since the same recording of the terminal session would also collect information on the time and success of compilations and assemblies, it could even be determined whether the programmer was able to rapidly remove source-code errors, or whether it took a large amount of compiler diagnostics until the source code could be considered correct. Thus, it appears that it should be relatively straightforward to implement a practically fully automatic Error Data Reporting Technique, but the question as to its psychological effect on the programmer must be resolved before it is introduced as a mandatory standard.

I.5 Program Documentation Standards

The subject of Program Documentation Standards is addressed in SPS Volume VII, Documentation Standards. The proposed modifications to the various DoD Standards will not present any problems for the Small Contractor who is generally used to supplying software according to one of these standards.

Structured Programming will make programs easier to read and understand and will therefore relieve some of the burden placed on Program Documentation which is usually almost exclusively relied on to provide sufficient understanding of the program structure to allow later maintenance, updating and modifications.

However, truly satisfactory Documentation Standards have so far not been developed. It is an amusing paradox that every programmer knows exactly the type of information he would need to maintain, modify or expand a program written by another programmer, while he consistently fails to provide that information on programs written by himself.

Suggestions for overcoming this situation are contained in the following.

I.5.1 Recommendations

Very powerful text processing and graphics software packages have recently become available which make it entirely feasible to produce extensive narrative text (in upper and lower case characters) as well as mathematical symbols and the Greek alphabet, such that very extensive comments could be inserted into the compilable code to fully explain every step taken in the program. In fact, it is even conceivable that the explanatory text is prepared first and the code which implements the intent of the narrative discussion is inserted later. The code with the full explanations could then serve as documentation, whereas, of course, only the actual code would be compiled. It might be an interesting R&D project to pursue this concept in which the documentation and the source code itself make up a single computer-compatible package.

Another recommendation is for the Government to supply examples of structured programs together with the desired Program Documentation in its optimum form.

It is usually much easier for a programmer to understand the intent of a documentation standard if he is presented with examples which he can readily relate to his own work.

METRIC SYSTEM

BASE UNITS:

Quantity	Unit	SI Symbol	Formula
length	metre	m	...
mass	kilogram	kg	...
time	second	s	...
electric current	ampere	A	...
thermodynamic temperature	kelvin	K	...
amount of substance	mole	mol	...
luminous intensity	candela	cd	...

SUPPLEMENTARY UNITS:

Quantity	Unit	SI Symbol	Formula
plane angle	radian	rad	...
solid angle	steradian	sr	...

DERIVED UNITS:

Quantity	Unit	SI Symbol	Formula
Acceleration	metre per second squared	...	m/s
activity (of a radioactive source)	disintegration per second	...	(disintegration)/s
angular acceleration	radian per second squared	...	rad/s
angular velocity	radian per second	...	rad/s
area	square metre	...	m
density	kilogram per cubic metre	...	kg/m
electric capacitance	farad	F	A·s/V
electrical conductance	siemens	S	A/V
electric field strength	volt per metre	...	V/m
electric inductance	henry	H	V·s/A
electric potential difference	volt	V	W/A
electric resistance	ohm		V/A
electromotive force	volt	V	W/A
energy	joule	J	N·m
entropy	joule per kelvin	...	J/K
force	newton	N	kg·m/s
frequency	hertz	Hz	(cycle)/s
illuminance	lux	lx	lm/m
luminance	candela per square metre	...	cd/m
luminous flux	lumen	lm	cd·sr
magnetic field strength	ampere per metre	...	A/m
magnetic flux	weber	Wb	V·s
magnetic flux density	tesla	T	Wb/m
magnetomotive force	ampere	A	...
power	watt	W	J/s
pressure	pascal	Pa	N/m
quantity of electricity	coulomb	C	A·s
quantity of heat	joule	J	N·m
radiant intensity	watt per steradian	...	W/sr
specific heat	joule per kilogram-kelvin	...	J/kg·K
stress	pascal	Pa	N/m
thermal conductivity	watt per metre-kelvin	...	W/m·k
velocity	metre per second	...	m/s
viscosity, dynamic	pascal-second	...	Pa·s
viscosity, kinematic	square metre per second	...	m/s
voltage	volt	V	W/A
volume	cubic metre	...	m
wavenumber	reciprocal metre	...	(wave)/m
work	joule	J	N·m

SI PREFIXES:

Multiplication Factors	Prefix	SI Symbol
1 000 000 000 000 = 10^{12}	tera	T
1 000 000 000 = 10^{9}	giga	G
1 000 000 = 10^{6}	mega	M
1 000 = 10^{3}	kilo	k
100 = 10^{2}	hecto*	h
10 = 10^{1}	deka*	da
0.1 = 10^{-1}	deci*	d
0.01 = 10^{-2}	centi*	c
0.001 = 10^{-3}	milli	m
0.000 001 = 10^{-6}	micro	μ
0.000 000 001 = 10^{-9}	nano	n
0.000 000 000 001 = 10^{-12}	pico	p
0.000 000 000 000 001 = 10^{-15}	femto	f
0.000 000 000 000 000 001 = 10^{-18}	atto	a

* To be avoided where possible.

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

AMERICAN REVOLUTION BICENTENNIAL 1776-1976